Generative Multi-Agent Behavioral Cloning

Eric Zhan
Caltech
ezhan@caltech.edu

Stephan Zheng
Caltech
stzheng@caltech.edu

Yisong Yue
Caltech
yyue@caltech.edu

Long Sha
STATS
lsha@stats.com

Patrick Lucey
STATS
plucey@stats.com

Abstract

We propose and study the problem of generative multi-agent behavioral cloning, where the goal is to learn a generative, i.e., non-deterministic, multi-agent policy from pre-collected demonstration data. Building upon advances in deep generative models, we present a hierarchical policy framework that can tractably learn complex mappings from input states to distributions over multi-agent action spaces by introducing a hierarchy with macro-intent variables that encode long-term intent. In addition to synthetic settings, we show how to instantiate our framework to effectively model complex interactions between basketball players and generate realistic multi-agent trajectories of basketball gameplay over long time periods. We validate our approach using both quantitative and qualitative evaluations, including a user study comparison conducted with professional sports analysts.[1]

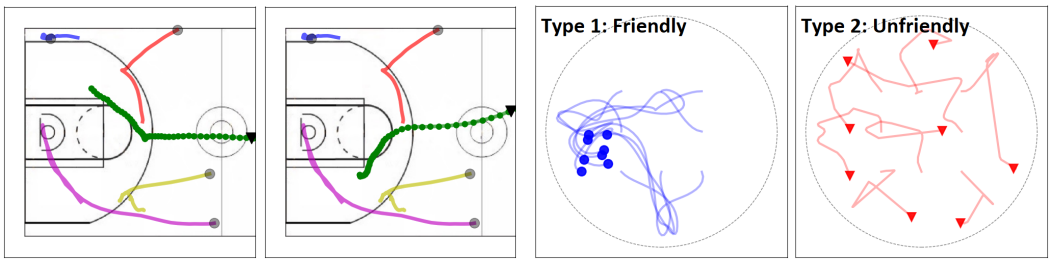
1 Introduction

The ongoing explosion of recorded tracking data is enabling the study of fine-grained behavior in many domains. Examples include sports Miller et al. [2014], Yue et al. [2014], Zheng et al. [2016], Le et al. [2017], video games Ross et al. [2011], video & motion capture Suwajanakorn et al. [2017], Taylor et al. [2017], Xue et al. [2016], navigation & driving Ziebart et al. [2009], Zhang and Cho [2017], Li et al. [2017], laboratory animal behaviors Johnson et al. [2016], Eyjolfsdottir et al. [2017], and tele-operated robotics Abbeel and Ng [2004], Lin et al. [2006].

In this paper, we are interested in imitating the behavior of multiple cooperating agents whose underlying policies are inherently non-deterministic and exhibit hierarchical structure. For example, Figure 1a depicts offensive player behavior in basketball in which players behave non-deterministically and the distribution over trajectories is multimodal. Figure 1b depicts a simplified Boids model from Reynolds [1987] for modeling animal schooling behavior in which the agents can be friendly or unfriendly. In both examples, the agents are highly coordinated as well as non-deterministic, and the space of multi-agent trajectories is naively exponentially large.

We thus study the problem of *generative multi-agent behavioral cloning*, where the policy maps input states to distributions over multi-agent action spaces. Unlike conventional behavioral cloning, the goal is not to perfectly mimic the demonstrations, but rather to recover their (latent) generating distribution. While there has been some work in multi-agent imitation learning Chernova and Veloso [2007], Le et al. [2017] and imitation learning with stochastic polices Ziebart et al. [2008], Ho and Ermon [2016], Li et al. [2017], no previous work has focused on learning generative polices as a core research direction, especially while simultaneously addressing multi-agent learning.

[1] The dataset and code for our experiments are available at https://github.com/ezhan94/gen-MA-BC.



(a) Offensive basketball players have multi-modal behavior (ball not shown). For instance, the green player (▼) moves to either the top-left or bottom-left.

(b) Two types of generated behaviors for 8 agents in Boids model. **Left**: Friendly blue agents group together. **Right**: Unfriendly red agents stay apart.

Figure 1: Examples of coordinated multimodal multi-agent behavior.

Contributions We present a hierarchical policy class that integrates recent advances in deep generative models into the behavioral cloning setting. Our framework introduces a hierarchy of macro-intent variables into the underlying deep graphical model that provides several advantages:

- It is straightforward to incorporate domain knowledge using macro-intents.
- It allows for conditional inference by grounding macro-intents to manipulate agent behavior.
- It allows for generating plans at multiple time scales, enabling effective long-term planning.
- It allows for tractably modeling of long-term coordination between multiple agents.
- It is compatible with existing variational methods for training deep generative models.

In addition to synthetic settings, we showcase our approach in an application on modeling team offense in basketball. We validate our approach both quantitatively and qualitatively, including a user study comparison with professional sports analysts, and show significant improvements over standard baselines. An interactive demo is available at `http://basketball-ai.com/`.

2 Related Work

Imitation Learning. One can roughly dichotomize imitation learning into: 1) passively learning to mimic batched pre-collected demonstrations Abbeel and Ng [2004], Ziebart et al. [2008], Ho and Ermon [2016], and 2) actively querying an oracle for feedback during learning Daumé et al. [2009], Ross et al. [2011]. Behavioral cloning Syed and Schapire [2008] belongs to the former and is often regarded as the simplest form of imitation learning. By learning via behavioral cloning, we focus our research on the modeling challenges that arise from learning generative multi-agent policies.

As mentioned in the introduction, there has been some prior work in multi-agent imitation learning and learning stochastic policies, but no previous work has focused on learning generative polices while simultaneously addressing generative and multi-agent imitation learning. For instance, experiments in Ho and Ermon [2016] all lead to highly peaked distributions, while Li et al. [2017] captures multimodal distributions by learning unimodal policies for a fixed number of experts. Hrolenok et al. [2017] bring up the issue of learning stochastic multi-agent behavior, but their solution relies on specifying a non-trivial feature function.

Long-term planning. Another issue that our work addresses is long-term planning. In this regard, the closest prior work is Zheng et al. [2016], which also reasoned over long sequences using macro-intents (which they call macro-goals). However, their approach was only for a single agent and used relatively simple stochastics. Beyond imitation learning, designing hierarchical policies is a topic of both historical and contemporary interest in reinforcement learning [Dayan and Hinton, 1993, Sutton et al., 1999, Kulkarni et al., 2016]. From that perspective, one can view our work as developing generative policies to capture complex non-deterministic behaviors.

Deep generative models. The study of deep generative models is an increasingly popular research area, due to their ability to inherit both the flexibility of deep learning and the probabilistic semantics of generative models. In general, there are two ways that one can incorporate stochastics into deep models. The first approach models an explicit distribution over actions in the output layer, e.g., via logistic regression [Chen et al., 2015, Oord et al., 2016a,b, Zheng et al., 2016, Eyjolfsdottir

et al., 2017]. The second approach uses deep neural nets to define a transformation from a simple distribution to one of interest [Goodfellow et al., 2014, Kingma and Welling, 2014, Rezende et al., 2014] and can more readily be extended to incorporate additional structure, such as a hierarchy of random variables [Ranganath et al., 2016] or dynamics [Johnson et al., 2016, Chung et al., 2015, Krishnan et al., 2017, Fraccaro et al., 2016]. Our framework can incorporate both variants.

3 Multi-Agent Behavioral Cloning

We formalize the environment as a Markov Decision Process (MDP) without reward for K (ordered) cooperative agents over a fixed time horizon T (described below). Our goal is to learn a generative multi-agent policy for this environment from expert demonstrations.

- Let $\mathbf{x}_t^k \in \mathcal{X}$, $\mathbf{a}_t^k \in \mathcal{A}$ denote the state, action of agent k at time t.
- Let $\tau_t = \{(\mathbf{x}_u, \mathbf{a}_u)\}_{1 \le u \le t} = \{(\mathbf{x}_u^k, \mathbf{a}_u^k)_{\text{agents } k}\}_{1 \le u \le t}$ denote the history of state-action pairs.
- Let $\pi_\theta(\mathbf{x}_t, \tau_{t-1})$ denote a multi-agent policy parametrized by θ that samples actions from the probability distribution $p_\theta(\mathbf{a}_t | \mathbf{x}_t, \tau_{t-1})$.
- Let $\mathcal{M}(\mathbf{x}_t, \mathbf{a}_t)$ denote a transition function for states: $\mathbf{x}_{t+1} \sim p_{\mathcal{M}}(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{a}_t)$.
- Let $\tau \sim \pi$ denote that τ was generated from policy π.
- Let $\mathcal{D}$ denote the collection of N expert demonstrations generated by expert policy π_E: $\mathcal{D} = \{\tau^{(i)} : \tau^{(i)} \sim \pi_E\}_{i=1}^N$.

Learning Objective Behavioral cloning uses supervised learning to find a policy that mimics the expert demonstrations $\mathcal{D}$ by solving an optimization problem with respect to a loss function ℓ:

$$\theta^* = \text{argmin}_\theta \, \mathbb{E}_{\tau \sim \pi_E} \left[\sum_{t=1}^{T} \ell\big(\mathbf{a}_t, \pi_\theta(\mathbf{x}_t, \tau_{t-1})\big) \right] \approx \text{argmin}_\theta \sum_{\tau \in \mathcal{D}} \sum_{t=1}^{T} \ell\big(\mathbf{a}_t, \pi_\theta(\mathbf{x}_t, \tau_{t-1})\big) \quad (1)$$

Simplifying assumptions. For many spatial environments, the transition $\mathcal{M}$ is typically deterministic: $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{a}_t$. As such, we can absorb $\mathcal{M}$ into the policy π_θ and predict $\mathbf{x}_t + \mathbf{a}_t$ directly. For example, the initial state $\mathbf{x}_1$ of the green player in Figure 1a is marked by ▼. The player's action $\mathbf{a}_1$ is to move left, which results in the next state $\mathbf{x}_2 = \mathbf{x}_1 + \mathbf{a}_1$. We can then simplify some notation:

- Since actions are now implicitly tied into the state, we can denote each demonstration as $\mathbf{x}_{\le T} = \{\mathbf{x}_t\}_{1 \le t \le T}$, and the history of states as $\tau_t = \mathbf{x}_{\le t}$.
- Similarly, the stochastic policy $\pi_\theta(\mathbf{x}_t, \tau_{t-1}) = \pi_\theta(\tau_t)$ now samples the next state directly from $p_\theta(\mathbf{x}_{t+1} | \tau_t)$. The policy is implicitly sampling an action.

For a stochastic policy that returns parameters of a distribution, the training loss ℓ is often the negative log-likelihood. This lets us re-write the objective in Eq. (1) as the following maximization problem:

$$\theta^* = \text{argmin}_\theta \sum_{\tau \in \mathcal{D}} \sum_{t=1}^{T} \ell\big(\mathbf{x}_t, \pi_\theta(\tau_{t-1})\big) = \text{argmax}_\theta \sum_{\tau \in \mathcal{D}} \sum_{t=1}^{T} \log p_\theta\big(\mathbf{x}_t | \tau_{t-1}\big). \quad (2)$$

Eq. (2) is also the objective for sequential generative models that maximize the log-likelihood of data $\mathcal{D} = \{\mathbf{x}_{\le T}\}$ by factorizing the joint distribution of the sequence:

$$\theta^* = \text{argmax}_\theta \sum_{\mathbf{x}_{\le T} \in \mathcal{D}} \log p_\theta(\mathbf{x}_{\le T}) = \text{argmax}_\theta \sum_{\mathbf{x}_{\le T} \in \mathcal{D}} \sum_{t=1}^{T} \log p_\theta(\mathbf{x}_t | \mathbf{x}_{<t}). \quad (3)$$

As we empirically verify in Section 5, models trained with Eq. (3) have difficulty learning representations of the data that generalize well over long time horizons. Our solution is to introduce a hierarchical structure of macro-intents as an effective means in learning low-dimensional (distributional) representations of the data that extend in time and space for multiple coordinating agents.

4 Generative Multi-Agent Policy Class

We now present our generative hierarchical multi-agent policy class that incorporates macro-intent in the higher layer of the hierarchy. We first assume conditional independence between the agent states $\mathbf{x}_t^k$ given history $\tau_{t-1} = \mathbf{x}_{<t}$. This lets us decompose the loss ℓ and policy π_θ in Eq. (2):

$$\theta^* = \text{argmin}_\theta \sum_{\tau \in \mathcal{D}} \sum_{t=1}^{T} \sum_{k=1}^{K} \ell\big(\mathbf{x}_t^k, \pi_\theta^k(\tau_{t-1})\big) = \text{argmax}_\theta \sum_{\tau \in \mathcal{D}} \sum_{t=1}^{T} \sum_{k=1}^{K} \log p_\theta^k(\mathbf{x}_t^k | \mathbf{x}_{<t}) \tag{4}$$

We can instantiate the agent-policies π_θ^k with any sequential generative model. In our experiments, we use a variational recurrent neural network (VRNN) Chung et al. [2015] as a base module, which is essentially a variational autoencoder (VAE) Kingma and Welling [2014] conditioned on the hidden state of a RNN. VRNNs introduce a stochastic latent variable $\mathbf{z}_t^k$ for each agent-policy:

$$\pi_\theta^k(\tau_{t-1}) \sim p_\theta^k(\mathbf{x}_t^k | \mathbf{x}_{<t}) = \varphi^k(\mathbf{z}_t^k, \mathbf{h}_{t-1}^k), \qquad \mathbf{h}_t^k = f^k(\mathbf{x}_t^k, \mathbf{z}_t^k, \mathbf{h}_{t-1}^k), \tag{5}$$

where φ^k maps to a distribution over states and f^k is a deterministic function such a GRU Cho et al. [2014]. Figure 3a depicts a graphical model diagram of the VRNN. During training, we maximize the evidence lower-bound (ELBO) of Eq. (4), which is just the VAE ELBO summed over each timestep t. Note below that q_ϕ is the inference function that approximates the posterior.[2]

$$\mathbb{E}_{q_\phi(\mathbf{z}_{\leq T} | \mathbf{x}_{\leq T})} \left[\sum_{t=1}^{T} \log p_\theta(\mathbf{x}_t \mid \mathbf{z}_{\leq T}, \mathbf{x}_{<t}) - D_{KL}\Big(q_\phi(\mathbf{z}_t \mid \mathbf{x}_{\leq T}, \mathbf{z}_{<t}) || p_\theta(\mathbf{z}_t \mid \mathbf{x}_{<t}, \mathbf{z}_{<t}) \Big) \right]. \tag{6}$$

Hierarchical policy class and macro-intents. Our overall policy is hierarchical and uses the intermediate layer to: 1) provide a tractable way to capture coordination between agents; 2) encode long-term intents of agents and enable long-term planning at a higher-level timescale; and 3) compactly represent some low-dimensional structure in an exponentially large multi-agent action space. Figure 2 illustrates macro-intents for two basketball players, which take the form of areas on the court. Upon reaching its macro-intent in the top-right, the blue player moves towards its next macro-intent in the bottom-left. Similarly, the green player moves towards its macro-intents from bottom-right to middle-left. Macro-intents are shared variables, so both players can see each other's macro-intent (i.e. where the other player is going).

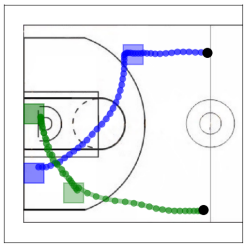


Figure 2: Macro-intents (boxes) for two players.

More generally, our modeling assumptions for macro-intents are:

- agent states $\{\mathbf{x}_t^k\}$ in an episode $[t_1, t_2]$ are conditioned on some fixed macro-intent $\mathbf{g}_t$,
- the start and end times $[t_1, t_2]$ of episodes can vary between trajectories,
- macro-intents change slowly over time relative to the agent states: $d\mathbf{g}_t/dt \ll 1$,
- and due to their reduced dimensionality, we can model (near-)arbitrary dependencies between macro-intents (e.g., coordination) via black box learning.

In general, macro-intents do not need to have a geometric interpretation. For example, macro-intents in the Boids model in Figure 1b can be a binary label indicating friendly vs. unfriendly behavior. The goal is for macro-intents to encode long-term intent and ensure that agents behave more cohesively.

Modeling macro-intents. Our hierarchical model uses an intermediate layer to model macro-intent variables, $\mathbf{g}_t$, so our agent-policies (Eq. (5)) become:

$$\pi_\theta^k(\tau_{t-1}) \sim p_\theta(\mathbf{x}_t^k | \mathbf{x}_{<t}) = \varphi^k(\mathbf{z}_t^k, \mathbf{h}_{t-1}^k, \mathbf{g}_t). \tag{7}$$

Figure 3b shows our hierarchical policy class, which generates macro-intents rather than using ground truth macro-intents. Here, we train an RNN-policy to sample macro-intents:

$$p(\mathbf{g}_t | \mathbf{g}_{<t}) = \varphi_g(\mathbf{h}_{g,t-1}, \mathbf{x}_{t-1}), \tag{8}$$

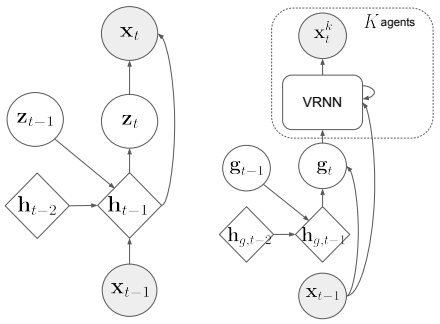
$$\mathbf{h}_{g,t} = f_g(\mathbf{g}_t, \mathbf{h}_{g,t-1}). \tag{9}$$

We condition the macro-intent policy on previous states $\mathbf{x}_{t-1}$ in Eq. (8) and generate next states by first sampling a macro-intent $\mathbf{g}_t$, and then sampling $\mathbf{x}_t^k$ conditioned on $\mathbf{g}_t$ (see Figure 3b).

[2] We refer to the appendix for an overview of VAEs and VRNNs.

Hierarchical learning. We can jointly learn our agent and macro-intent policies by maximizing the VRNN objective from Eq (6) conditioned on the shared $\mathbf{g}_t$ variables. However, we found in practice that this does not lead to the model learning meaningful macro-intents. Instead, we train the agent and macro-intent policies independently, where the macro-intent policy is learned via supervised learning by maximizing the log-likelihood of macro-intent labels.

One can collect macro-intent labels for training in a variety of ways. While having expert labels is ideal, we show that it is straightforward to generate weak labels using simple heuristics. This allows us to incorporate domain knowledge into the model. For instance, setting macro-intents to be areas on the court in basketball incorporates the idea that players aim to set up specific formations. Similar techniques have been employed in other weak supervision settings, e.g., Ratner et al. [2016, 2018].



(a) VRNN (b) Our model

Figure 3: Depicting VRNN and our model. Circles are stochastic and diamonds are deterministic. macro-intent $\mathbf{g}_t$ is shared across agents. In principle, any generative model can be used in our framework.

5 Experiments

We first apply our approach on generating offensive team basketball gameplay (team with possession of the ball), and then on a synthetic Boids model dataset. We present both quantitative and qualitative experimental results. Our quantitative results include a user study comparison with professional sports analysts, who significantly preferred basketball rollouts generated from our approach to standard baselines. Examples from the user study and videos of rollouts are in the supplementary material. Our qualitative results demonstrate the ability of our approach to generate high-quality rollouts under various conditions. An interactive demo is available at `http://basketball-ai.com/`.

5.1 Experimental Setup for Basketball

Training data. Each demonstration in our data contains trajectories of $K = 5$ players on the left half-court, recorded for $T = 50$ timesteps at 6 Hz. The offensive team has possession of the ball for the entire sequence. $\mathbf{x}_t^k$ are the coordinates of player k at time t on the court (50×94 feet). We normalize and mean-shift the data. Players are ordered based on their relative positions, similar to the role assignment in Lucey et al. [2013]. Overall, there are 107,146 training and 13,845 test examples.

For simplicity, we ignore the defensive players to focus on capturing the coordination of the offensive team. In addition, the defense is usually reactionary whereas the offense takes the initiative and tends to have more multimodal behavior. In principle, we can provide the defensive positions as conditional input for our model and update the defensive positions using methods such as Le et al. [2017]. We also ignore the ball since the ball dynamics are difficult to learn (e.g. oscillations indicate dribbling while straight lines indicate passing). We leave the task of modeling the ball for future work.

Weak macro-intent labels. We extract weak macro-intent labels $\hat{\mathbf{g}}_t^k$ for each player k as done in Zheng et al. [2016]. We segment the left half-court into a 10×9 grid of 5ft $\times$5ft cells. The weak macro-intent $\hat{\mathbf{g}}_t^k$ at time t is a 1-hot encoding of dimension 90 of the next cell in which player k is stationary (speed $\|\mathbf{x}_{t+1}^k - \mathbf{x}_t^k\|_2$ below a threshold). The shared macro-intent $\mathbf{g}_t$ is the concatenation of individual macro-intents. Macro-intents change slowly over time relative to player positions (see Figure 2). Figure 4 shows the distribution of extracted weak macro-intent labels for each player.

Model details. We model each latent variable $\mathbf{z}_t^k$ as a multivariate Gaussian with diagonal covariance of dimension 16. All policies are implemented with memory-less 2-layer fully-connected neural networks with a hidden layer of size 200. Our agent-policies sample from a multivariate Gaussian with diagonal covariance while our macro-intent policies sample from a multinomial distribution over the macro-intents. All hidden states ($\mathbf{h}_{g,t}, \mathbf{h}_t^1, \ldots \mathbf{h}_t^K$) are modeled with 200 2-layer GRU memory

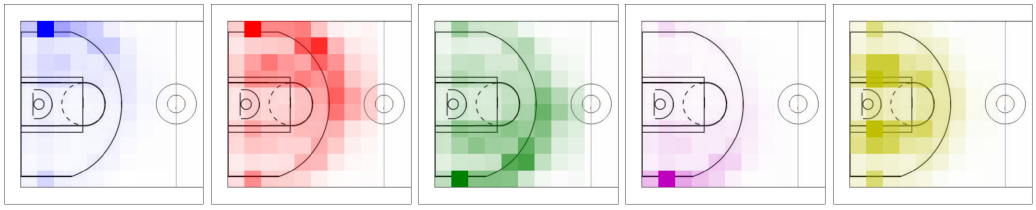


Figure 4: Distribution of weak macro-intent labels extracted for each player from the training data. Color intensity corresponds to frequency of macro-intent label. Players are ordered by their relative positions on the court, which can be seen from the macro-intents.

MODEL	BASKETBALL	BOIDS
RNN-GAUSS	1931	2414
VRNN-SINGLE	$\geq$ 2302	$\geq$ 2417
VRNN-INDEP	$\geq$ 2360	$\geq$ 2385
OURS	$\geq$ **2362**	$\geq$ **2428**

Table 1: Average log-likelihoods per test sequence. "$\geq$" indicates ELBO of log-likelihood. Our hierarchical model achieves higher log-likelihoods than baselines for both datasets.

VS. MODEL	WIN/TIE/LOSS	AVG GAIN
VS. VRNN-SINGLE	25/0/0	0.57
VS. VRNN-INDEP	15/4/6	0.23

Table 2: Basketball preference study results. Win/Tie/Loss indicates how often our model is preferred over baselines (25 comparisons per baseline). Gain is computed by scoring +1 when our model is preferred and -1 otherwise. Results are 98% significant using a one-sample t-test.

cells each. We maximize the log-likelihood/ELBO with stochastic gradient descent using the Adam optimizer Kingma and Ba [2014] and a learning rate of 0.0001.

Baselines. We compare our approach with 3 baselines that do not use a hierarchy of macro-intents:

1. **RNN-gauss:** RNN without latent variables using 900 2-layer GRU cells for the hidden state.
2. **VRNN-single:** VRNN in which we concatenate all player positions together ($K = 1$) with 900 2-layer GRU cells for the hidden state and a 80-dimensional latent variable.
3. **VRNN-indep:** VRNN for each agent with 250 2-layer GRU cells for the hidden states and 16-dimensional latent variables. We also provide the previous positions of all players as conditional input for each policy, so Eq. (5) becomes $p_\theta^k(\mathbf{x}_t^k|\mathbf{x}_{<t}) = \varphi^k(\mathbf{z}_t^k, \mathbf{h}_{t-1}^k, \mathbf{x}_{t-1})$.

5.2 Quantitative Evaluation for Basketball

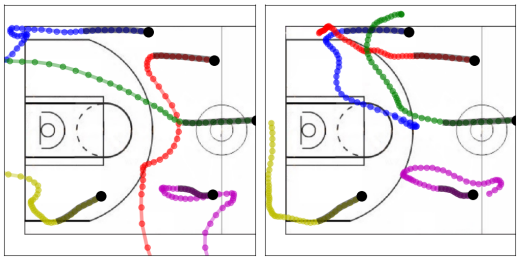
Log-likelihood. Table 1 reports the average log-likelihoods on the test data. Our approach outperforms RNN-gauss and VRNN-single and is comparable with VRNN-indep. However, higher log-likelihoods do not necessarily indicate higher quality of generated samples Theis et al. [2015]. As such, we also conduct a human preference study to assess the relative quality of generated rollouts.

Human preference study. We recruited 14 professional sports analysts as judges to compare the quality of rollouts. Each comparison animates two rollouts, one from our model and another from a baseline. Both rollouts are burned-in for 10 timesteps with the same ground-truth states from the test set, and then generated for the next 40 timesteps. Judges decide which of the two rollouts looks more realistic. Example comparisons are in the supplementary material.

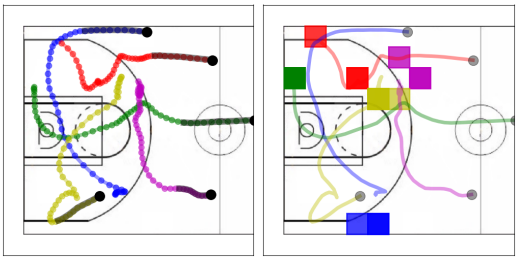
Table 2 shows the results from the preference study. We tested our model against two baselines, VRNN-single and VRNN-indep, with 25 comparisons for each. All judges preferred our model over the baselines with 98% statistical significance. These results suggest that our model generates rollouts of significantly higher quality than the baselines.

5.3 Qualitative Evaluation of Generated Rollouts for Basketball

We next conduct a qualitative visual inspection of rollouts. Figure 5 shows rollouts generated from VRNN-single, VRNN-indep, and our model by sampling states for 40 timesteps after an initial burn-in period of 10 timesteps with ground-truth states from the test set. An interactive demo to generate more rollouts from our hierarchical model can be found at: `http://basketball-ai.com/`.



(a) Baseline rollouts of representative quality. **Left**: VRNN-single. **Right**: VRNN-indep. Common problems in baseline rollouts include players moving out of bounds or in the wrong direction. Players do not appear to behave cohesively as a team.



(b) **Left**: Rollout from our model. All players remain in bounds. **Right**: Corresponding macro-intents for left rollout. Macro-intent generation is stable and suggests that the team is creating more space for the blue player (perhaps setting up an isolation play).

Figure 5: Rollouts from baselines and our model starting from black dots, generated for 40 timesteps after an initial burn-in period of 10 timesteps (marked by dark shading). An interactive demo of our hierarchical model is available at: `http://basketball-ai.com/`.

Common problems in baseline rollouts include players moving out of bounds or in the wrong direction (Figure 5a). These issues tend to occur at later timesteps, suggesting that the baselines do not perform well over long horizons. One possible explanation is due to compounding errors Ross et al. [2011]: if the policy makes a mistake and deviates from the states seen during training, it is likely to make more mistakes in the future, thus leading to poor generalization.

On the other hand, generated rollouts from our model are more robust to the types of errors made by the baselines (Figure 5b). Generated macro-intents also allow us to intepret the intent of each individual player as well as a global team strategy that all players execute cohesively (e.g. setting up a specific formation on the court). We highlight that our model learns a multimodal generating distribution, as repeated rollouts with the same burn-in result in a dynamic range of generated trajectories, as seen in Figure 6 Left. Furthermore, Figure 6 Right demonstrates that grounding macro-intents during generation instead of sampling them allows us to control agent behavior.

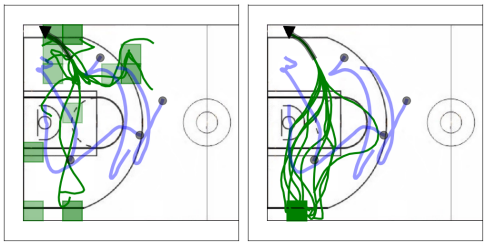


Figure 6: 10 rollouts of the green player (▼). A burn-in period of 20 timesteps is applied. Blue trajectories are the other players and (•) indicates initial positions. **Left**: The model generates macro-intents. **Right**: We ground the macro-intents at the bottom-left. In both, we observe a multi-modal distribution of trajectories.

5.4 Synthetic Experiments: Boids Model

To illustrate the generality of our approach, we apply our model to a simplified version of the Boids model Reynolds [1987] that produces realistic trajectories of schooling behavior. We generate trajectories for 8 agents for 50 frames. The agents start in fixed positions around the origin with initial velocities sampled from a unit Gaussian. Each agent's velocity is then updated at each timestep:

$$\mathbf{v}_{t+1} = \beta \mathbf{v}_t + \beta(c_1 \mathbf{v}_{\text{coh}} + c_2 \mathbf{v}_{\text{sep}} + c_3 \mathbf{v}_{\text{ali}} + c_4 \mathbf{v}_{\text{ori}}), \tag{10}$$

where $\mathbf{v}_{\text{coh}}$ is the normalized cohesion vector towards the center of an agent's local neighborhood (other agents within some radius), $\mathbf{v}_{\text{sep}}$ is the normalized vector away from an agent's close neighborhood (smaller radius than for $\mathbf{v}_{\text{coh}}$), $\mathbf{v}_{\text{ali}}$ is the average velocity of other agents in a local neighborhood, and $\mathbf{v}_{\text{ori}}$ is the normalized vector towards the origin. We fix c_2, c_3, and c_4 to be positive constants, but we randomly sample the sign of c_1 before generating a new trajectory. This produces two distinct types of behaviors: *friendly agents* ($c_1 > 0$) that like to group together, and *unfriendly agents* ($c_1 < 0$) that like to stay apart (see Figure 1b). Lastly, we introduce more stochasticity into the model by uniformly sampling β every 10 frames in a range about 1.

We train our model and baselines to generate the actions of agents using 32,768 training and 8,192 test trajectories. We use the sign of c_1 as our macro-intents, which indicates the type of behavior. Note that unlike our macro-intents for the basketball dataset, these macro-intents are simpler and

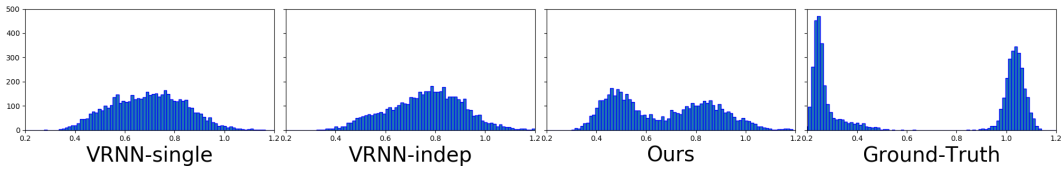


Figure 7: Synthetic Boids experiments. Showing histograms (horizontal axis: distance; vertical: counts) of average distance to an agent's closest neighbor in 5000 roll-outs. Our hierarchical model more closely captures the two distinct modes for friendly (small distances, left peak) vs. unfriendly (large distances, right peak) behavior compared to baselines, which do not learn to distinguish them.

have no geometric interpretation. All models have similar average log-likelihoods on the test set in Table 1, but our hierarchical model can capture the true generating distribution much better than the baselines. For example, we compute the average distance to an agent's closest neighbor in generated trajectories from all models and the ground-truth and plot the histograms in Figure 7. We see that this statistic for the ground-truth has two distinct modes for friendly (small distances, left peak) vs. unfriendly (large distances, right peak) behavior. Our model more closely captures these two modes whereas the baselines fail to distinguish them.

5.5 Analysis of Hierarchical Policy Class

Output distribution for states. The agent-policies in all our models (including baselines) sample from a multivariate Gaussian with diagonal covariance. We also experimented with sampling from a mixture of 2, 3, 4, and 8 Gaussian components, but discovered that the models would always learn to assign all the weight on a single component and ignore the others. The variance of the active component is also very small. This is intuitive because sampling with a large variance at every timestep would result in noisy trajectories and not the smooth ones that we see in Figures 5, 6.

Choice of macro-intent policy model. We chose to model our macro-intent policy in Eq. (8-9) with an RNN. In principle, we can also use more expressive models, like a VRNN, to model macro-intent policies over richer macro-intent spaces. In our case, we found that an RNN was sufficient in capturing the distribution of macro-intents shown in Figure 4. The RNN learns multinomial distributions over macro-intents that are peaked at a single macro-intent and relatively static through time, which is consistent with the behavior of macro-intents that we extracted from the data. Latent variables in a VRNN had minimal effect on the multinomial distribution.

Hidden state for macro-intent policy model. Specifically for the basketball dataset, we defined macro-intents $\mathbf{g}_t$ to be the concatenation of individual macro-intents $\mathbf{g}_t^k$ for each player. For our macro-intent policy, we compared a RNN model with a shared hidden state in Eq. (9), with a RNN model with independent hidden states. Intuitively, we expect the shared hidden state model to be better at capturing coordination.

For instance, good coordination in basketball corresponds to diverse macro-intents, i.e. players should choose different regions on the court as long-term goals. To provide a more quantitative comparison, we computed the frequency at which two or more players had the same individual macro-intent $\mathbf{g}_t^k$ at a given time, with the assumption that coordinated players do not have coinciding macro-intents very often. In the training data, 5.7% of all timesteps had coinciding macro-intents. In 10,000 rollouts from our macro-intent policy, 8.5% and 15.2% of all timesteps had coinciding macro-intents for the shared and independent hidden state models respectively. As a result, we used a RNN with a shared hidden state to model the macro-intent policy.

6 Discussion

The macro-intents used in our experiments are relatively simple. For instance, rather than simply using location-based macro-intents, we can also incorporate interactions such as "pick and roll". Another future direction is to explore how to adapt our method to different domains, e.g., learning a macro-intent representing "argument" for a dialogue between two agents, or a macro-intent representing

"refrain" for music generation for "coordinating instruments" Thickstun et al. [2017]. We have shown that weak macro-intent labels can be effectively extracted using simple domain-specific heuristics. An interesting direction is to incorporate multiple heuristics, each viewed as noisy realizations of the true macro-intents, similar to Ratner et al. [2016, 2018]. One could also consider unsupervised learning of macro-intents. For example, in Vezhnevets et al. [2016] an agent learns plans of future actions and is penalized for changing its plan. Such concepts could be incorporated in our method.

Acknowledgements

This research is supported in part by NSF #1564330, NSF #1637598, and gifts from Bloomberg, Activision/Blizzard and Northrop Grumman. Dataset was provided by STATS: `https://www.stats.com/data-science/`.

References

Andrew Miller, Luke Bornn, Ryan Adams, and Kirk Goldsberry. Factorized point process intensities: A spatial analysis of professional basketball. In *ICML*, 2014.

Yisong Yue, Patrick Lucey, Peter Carr, Alina Bialkowski, and Iain Matthews. Learning fine-grained spatial models for dynamic sports play prediction. In *ICDM*. IEEE, 2014.

Stephan Zheng, Yisong Yue, and Patrick Lucey. Generating long-term trajectories using deep hierarchical networks. In *NIPS*, 2016.

Hoang Minh Le, Yisong Yue, Peter Carr, and Patrick Lucey. Coordinated multi-agent imitation learning. In *ICML*, 2017.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. No-regret reductions for imitation learning and structured prediction. In *AISTATS*, 2011.

Supasorn Suwajanakorn, Steven M Seitz, and Ira Kemelmacher-Shlizerman. Synthesizing obama: learning lip sync from audio. *ACM Transactions on Graphics (TOG)*, 36(4):95, 2017.

Sarah Taylor, Taehwan Kim, Yisong Yue, Moshe Mahler, James Krahe, Anastasio Garcia Rodriguez, Jessica Hodgins, and Iain Matthews. A deep learning approach for generalized speech animation. In *SIGGRAPH*, 2017.

Tianfan Xue, Jiajun Wu, Katherine Bouman, and Bill Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *NIPS*, 2016.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Human behavior modeling with maximum entropy inverse optimal control. In *AAAI Human Behavior Modeling*, 2009.

Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end autonomous driving. In *AAAI*, 2017.

Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In *NIPS*, 2017.

Matthew Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. In *NIPS*, 2016.

Eyrun Eyjolfsdottir, Kristin Branson, Yisong Yue, and Pietro Perona. Learning recurrent representations for hierarchical behavior modeling. In *ICLR*, 2017.

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, 2004.

Henry C Lin, Izhak Shafran, David Yuh, and Gregory D Hager. Towards automatic skill evaluation: Detection and segmentation of robot-assisted surgical motions. *Computer Aided Surgery*, 11(5): 220–230, 2006.

Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21(4):25–34, August 1987. ISSN 0097-8930.

Sonia Chernova and Manuela Veloso. Multiagent collaborative task learning through imitation. In *International Symposium on Imitation in Animals and Artifacts*, 2007.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*. Chicago, IL, USA, 2008.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *NIPS*, 2016.

Hal Daumé, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. In *NIPS*, 2008.

Brian Hrolenok, Byron Boots, and Tucker Balch. Sampling beats fixed estimate predictors for cloning stochastic behavior in multiagent systems. In *AAAI*, 2017.

Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In *NIPS*, 1993.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *NIPS*, 2016.

Liang-Chieh Chen, Alexander Schwing, Alan Yuille, and Raquel Urtasun. Learning deep structured models. In *ICML*, 2015.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016a.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016b.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *ICML*, 2016.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *NIPS*, 2015.

Rahul G. Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *AAAI*, 2017.

Marco Fraccaro, Søren Kaae Sø nderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *NIPS*, 2016.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014.

Alexander Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. 2016.

Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. 2018.

Patrick Lucey, Alina Bialkowski, Peter Carr, Stuart Morgan, Iain Matthews, and Yaser Sheikh. Representing and discovering adversarial team behaviors using player roles. In *CVPR*, 2013.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. *ArXiv e-prints*, November 2015.

John Thickstun, Zaid Harchaoui, and Sham Kakade. Learning features of music from scratch. In *ICLR*, 2017.

Alexander Vezhnevets, Volodymyr Mnih, John Agapiou, Simon Osindero, Alex Graves, Oriol Vinyals, and Koray Kavukcuoglu. Strategic attentive writer for learning macro-actions. *CoRR*, abs/1606.04695, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Appendix

We give a brief overview of deep generative models used to instantiate our policy class in our experiments. In particular, we review recurrent neural networks (RNNs), variational autoencoders (VAEs) and variational RNNs (VRNNs).

Recurrent neural networks. A RNN models the conditional probabilities in Eq. (3) with a hidden state $\mathbf{h}_t$ that summarizes the information in the first $t-1$ timesteps:

$$p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}) = \varphi(\mathbf{h}_{t-1}), \qquad \mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1}), \tag{11}$$

where φ maps the hidden state to a probability distribution over states and f is a deterministic function such as LSTMs Hochreiter and Schmidhuber [1997] or GRUs Cho et al. [2014]. RNNs with simple output distributions often struggle to capture highly variable and structured sequential data. Recent work in sequential generative models aim to address this issue by injecting stochastic latent variables into the model and using amortized variational inference to infer the latent variables from the data.

Variational Autoencoders. A variational autoencoder (VAE) Kingma and Welling [2014] is a generative model for non-sequential data that injects latent variables $\mathbf{z}$ into the joint distribution $p_\theta(\mathbf{x}, \mathbf{z})$ and introduces an inference network parametrized by ϕ to approximate the posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$. The learning objective is to maximize the evidence lower-bound (ELBO) of the log-likelihood with respect to the model parameters θ and ϕ:

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - D_{KL}(q_\phi(\mathbf{z} \mid \mathbf{x})||p_\theta(\mathbf{z})) \tag{12}$$

The first term is known as the reconstruction term and can be approximated with Monte Carlo sampling. The second term is the Kullback-Leibler divergence between the approximate posterior and the prior, and can be evaluated analytically (i.e. if both distributions are Gaussian with diagonal covariance). The inference model $q_\phi(\mathbf{z} \mid \mathbf{x})$, generative model $p_\theta(\mathbf{x} \mid \mathbf{z})$, and prior $p_\theta(\mathbf{z})$ are often implemented with neural networks.

Variational RNNs. VRNNs combine VAEs and RNNs by conditioning the VAE on a hidden state $\mathbf{h}_t$ (see Figure 3a):

$$p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}) = \varphi_{\text{prior}}(\mathbf{h}_{t-1}) \qquad \text{(prior)} \tag{13}$$

$$q_\phi(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t}) = \varphi_{\text{enc}}(\mathbf{x}_t, \mathbf{h}_{t-1}) \qquad \text{(inference)} \tag{14}$$

$$p_\theta(\mathbf{x}_t|\mathbf{z}_{\leq t}, \mathbf{x}_{<t}) = \varphi_{\text{dec}}(\mathbf{z}_t, \mathbf{h}_{t-1}) \qquad \text{(generation)} \tag{15}$$

$$\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{z}_t, \mathbf{h}_{t-1}). \qquad \text{(recurrence)} \tag{16}$$

VRNNs are also trained by maximizing the ELBO, which in this case can be interpreted as the sum of the ELBOs over each timestep of the sequence:

$$\mathbb{E}_{q_\phi(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})}\left[\sum_{t=1}^{T} \log p_\theta(\mathbf{x}_t \mid \mathbf{z}_{\leq T}, \mathbf{x}_{<t}) - D_{KL}\Big(q_\phi(\mathbf{z}_t \mid \mathbf{x}_{\leq T}, \mathbf{z}_{<t})||p_\theta(\mathbf{z}_t \mid \mathbf{x}_{<t}, \mathbf{z}_{<t})\Big)\right] \tag{17}$$

Note that the prior distribution of latent variable $\mathbf{z}_t$ depends on the history of states and latent variables (Eq. (13)). This temporal dependency of the prior allows VRNNs to model complex sequential data like speech and handwriting Chung et al. [2015].